Handle User Inputs and Forms

Making Requests from HTML Pages to PHP APIs (Forms)

- Example: Communication through a form and POST method
  - Backend Code: submit.php
  - Frontend Code: test2.html

Example: Communication through a form and POST method

Frontend (`test2.html`) ⟷ **Backend** (`submit.php`)

- In this section, we use a form and POST to request JavaScript.
- PHP server handles the POST request to return a JSON string as a response.

Backend Code: submit.php

<?php switch ($method) { case 'POST': // Handle POST parameters $name = $_POST['name'] ?? ''; $email = $_POST['email'] ?? ''; $info = ['name' => $name, 'email' => $email]; sendResponse($info, 'Response from POST request'); break; Default: sendError('Method Not Supported', 404); break; } ?>

Frontend Code: test2.html

- HTML form action specifies the PHP server endpoint to access (submit.php) with POST method (by making a post request).

```
<form action="submit.php" method="post">
  Name: <input type="text" name="name"><br>
  Email: <input type="email" name="email"><br>
  <input type="submit" value="Submit">
</form>
```

JavaScript

- The previous form has a listener when the `submit` button is clicked.
- It invokes a lambda (anonymous) function that does three actions.

```
document.querySelector('form').addEventListener('submit', async function (event) {
   event.preventDefault(); // Prevent the default form submission
   // 1. make a request
   // 2. get the response
   // 3. display results
});
```

1. Make a request

```
const form = event.target;
const formData = new FormData(form);
const response = await fetch(form.action, {
  method: 'POST',
  body: formData
});
```

- event.target → the <form> element that was submitted
- new FormData(form) → collects all form inputs into a FormData object
- form.action → the URL where the form should be submitted `<form action="submit.php" method="post">`
- fetch(...) → sends a request:
  - method: 'POST' → use POST request
  - body: formData → sends fields/files as multipart/form-data

This is the POST request made from JavaScript.

```
POST /submit.php HTTP/1.1
Host: localhost:3000
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Length: [calculated automatically]

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="name"

Alice
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="email"

alice@example.com
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

2. get the response

PHP server parses the request and make a response.

```php
case 'POST':
    // Handle POST parameters
    $name = $_POST['name'] ?? '';
    $email = $_POST['email'] ?? '';
    $info = [
        'name' => $name,
        'email' => $email
    ];
    sendResponse($info, 'Response from POST request');
```

```
const data = await response.json();
```

3. display results

```
try {
  document.getElementById('result').textContent =
    JSON.stringify(data, null, 2);
} catch (error) {
  document.getElementById('result').textContent =
    'Error: ' + error.message;
}
```

Complete JavaScript code

```
    document.querySelector('form').addEventListener('submit', async function (event) {
      event.preventDefault(); // Prevent the default form submission

      const form = event.target;
      const formData = new FormData(form);

      try {
        const response = await fetch(form.action, {
          method: 'POST',
          body: formData
        });

        const data = await response.json();

        // Display the result
        document.getElementById('result').textContent =
          JSON.stringify(data, null, 2);
      } catch (error) {
        document.getElementById('result').textContent =
          'Error: ' + error.message;
      }
    });
  </script>
```